APPENDIX A

Marked-up version of amended paragraphs

[0013] Fig. 1 shows a generation of modified program code to enable use of a storage area for a predicted load value, according to one embodiment of the invention. While the exemplary embodiment of Fig. 1 shows a [conversion] modification of source code to source code, other embodiments may include other types of [conversion] modification (e.g., source code to object code). Fig. 2 shows a flow chart of a method performed by a compiler to generate the modified code of Fig. 1, according to one embodiment of the invention. While in the exemplary embodiment the modification is performed by a compiler, alternative embodiments may have this modification performed by other entities (e.g., an emulator). While the following text describes Figs. 1 and 2 with respect to one another, the code of Fig. 1 may be generated without the method of Fig. 2 and the method of Fig. 2 may generate code other than that shown in Fig. 1. The particular instruction labels and instruction formats shown are for example only - it will be appreciated that other labels and formats may also be used.

[0017] [In Fig. 1, the compiler inserts] Fig. 1 shows inserting a check-predicted-value (chkpv) instruction at some point after the ldpv instruction. When executed, the chkpv instruction checks whether execution of the ldpv instruction created a cache miss and whether the predicted load value was correct. Efficient performance of the modified code during execution may depend on placement of the chkpv instruction. At block 260 of Fig. 2, the compiler determines a suitable insertion point and inserts the chkpv instruction into the modified code at that point at block 270. The chkpv instruction may be configured to branch to repair code under conditions described later.

At block 280, the compiler generates the repair code shown in Fig. 1, before continuing the compile operation at block 290. While in one operation, the chkpv instruction is inserted sufficiently far after the ldpv instruction that the cache line (and therefore the actual load value) is likely to be available by the time the chkpv instruction is executed, alternate embodiments may use other criteria to determine where to place the chkpv instruction.

[0025] ID - An identifier that [uniquely] identifies the particular load instruction or instruction sequence. Although in some embodiments other fields may be used for this purpose (e.g., see the description for the Address and Register No. fields below), the ID field may be used to avoid ambiguity if the other fields might contain duplicate information for multiple entries. In one embodiment, the identifier is placed in the ldpv and/or chkpv instructions at compile time. In another embodiment, the identifier is assigned at execution time.

[0044] If the comparison at block 565 determines the actual load value is the same as the predicted load value, then the previously-executed ldpv and subsequent load-dependent instructions were executed using correct values, and re-execution of those instructions may not be necessary. If the table entry corresponding to the associated ldpv instruction is no longer needed, in one embodiment that entry may be removed or invalidated at block 575. In another embodiment, the old entry may be left in the table until it is pushed out by a new entry. Execution may continue with the subsequent code at block 580.

Marked-up version of amended claims

1. (Amended once) An apparatus, comprising:

execution logic to execute a load instruction with a predicted load value; and

a [table] storage structure [coupled to the execution logic and] having a first field to store the predicted load value, the [table] storage structure to be used in repairing a mis-prediction of the predicted load value after a cache miss caused by a request of an actual load value.

3. (Amended once) The apparatus of claim 1, wherein:

the [table] storage structure includes a miss status holding register.

4. (Amended once) The apparatus of claim 1, wherein:

the [table] storage structure includes a second field to indicate a destination register for the load instruction.

5. (Amended once) The apparatus of claim 1, wherein:

the [table] storage structure includes a second field to store an address identifying a location of the actual load value.

6. (Amended once) A system, comprising:

a main memory;

a cache memory coupled to the main memory;

instruction execution logic [coupled to the cache memory] to execute a load instruction with an actual load value if a request of the actual load value

results in a cache hit in the cache memory and to execute the load instruction with a predicted load value if the request of the actual load value results in a cache miss in the cache memory; and

a [table] storage structure coupled to the instruction execution logic and having a first field to store the predicted load value if the request of the actual load value results in the cache miss.

7. (Amended once) The system of claim 6, wherein:

the [table] storage structure includes a miss status holding register.

8. (Amended once) The system of claim 6, wherein:

the instruction execution logic [is further] includes logic to execute a check instruction to compare the predicted load value from the first field with the actual load value.

9. (Amended once) The system of claim 8, wherein:

the instruction execution logic [is further] includes logic to branch to repair code if the predicted load value is different than the actual load value.

10. (Amended once) The system of claim 6, wherein:

the [table] storage structure is to not store the predicted load value if the request for the actual load value results in the cache hit.

11. (Amended once) An apparatus, comprising:

an instruction set in a processor[to execute an] , the instruction set including

a first set of one or more instructions to load a predicted load value and to place the predicted load value in a table in response to an attempt to load an actual load value resulting in a cache miss; and

a second set of one or more instructions to compare the predicted load value from the table with the actual load value and branch to repair code if the actual load value is different than the predicted load value.